리눅스 파일시스템을 위한 로그 기반 침입 복구 모듈의 구현*

이재국° 김형식
충남대학교 컴퓨터공학과
{empire, hskim}@cs.cnu.ac.kr

Implementation of A Log-Based Intrusion Recovery Module for Linux File System

Jae-Kook Lee, Hyong-Shik Kim
Dept. of Computer Science and Engineering, Chungnam National University

요 약

사용자는 침입이 있더라도 항상 신뢰성 있는 정보를 획득하길 원하기 때문에 침입에 의하여 파일이 훼손되는 경우에도 사용자에게 투명한 방법으로 복구할 수 있는 방법이 필요하다.

본 논문에서는 리눅스 기반의 파일 시스템에서 변경이 일어날 때마다 로그 형태로 저장된 로그 파일을 이용하여 침입에 의하여 훼손된 부분을 복구하기위한 모듈을 구현하고, 시험을 통하여 로그 기반 침입 복구 모듈을 적재한 시스템에서 로그를 관리하기 위해 발생하는 오버헤드를 분석한다.

1. 서 론

악의 있는 해커들은 컴퓨터 바이러스, 트로이목마, 웜 등의 악성 프로그램을 이용하여 시스템에 침입하고 파일을 변경(추가, 수정, 삭제)함으로써 일반 사용자로 하여금 올바른 정보를 받아 보지 못하게 파일을 훼손한다. 이런 악성 프로그램으로 파일 시스템이 훼손되는 것을 막기 위해 침입탐지 시스템, 침입차단시스템의 보안 솔루션이 제안되었다. 그러나 이들 보안 솔루션의 한계점으로 인하여 파일 시스템이 훼손될 경우 이를 극복하기 위한 방법으로 피해 파일 시스템을 침입 이전으로 복구하는 침입 복구 기술의 필요성이 제기되었다.

로그 기반 침입 복구 기법은 리눅스 파일시스템에서 파일에 변경이 발생하면 변경된 데이터에 대한 로그를 사용자에게 투명한 방법으로 기록하도록 하였다[1]. 본 논문에서는 제안된 로그 기반의 침입 복구 기법을 커널 전체를 재 컴파일하지 않고 동적으로 로드할 수 있도록 리눅스 커널 모듈 프로그래밍을 이용하여 구현하고, 특정 파일의 데이터 변경시 로그를 관리하기 위하여 발생하는 오버헤드를 분석한다.

2. 로그 기반 침입 복구 기법

본 절에서는 리눅스 파일시스템에서 특정파일의 변경시 침입 복구를 위해 사용자에게 투명한 방법으로 저장되는 로그의 구조를 보이고, 로그를 저장하기위한 기법과 신뢰된 로그에 대한 퍼지 기법, 침입에 의해 파일이 훼손된 경우 복구하기 위한 기법을 설명한다.

2.1 로그구조

침입 복구 모듈에서 침입 복구를 위해 사용될 로그파일의 구조는 그림 1과 같이 크게 두 부분으로 구분된다.

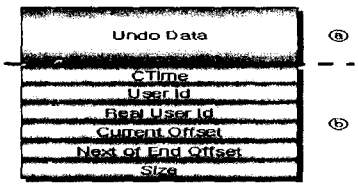


그림 1: 침입 복구를 위한 로그구조

ⓐ 부분은 원본파일의 변경되기 전의 내용을 기록하는 부분이고, ⓑ 부분은 원본파일의 변경된 내용에 대한 메타정보와 피해평가나 복구할 때 이용될 시간 정보와 역할 정보를 나타내는 로그헤더부분이다.

2.2 로그 저장 기법

그림 2의 (a)와 같이 특정 파일이 사용자에 의해 변경이 일어날 때 침입복구모듈은 사용자에게 투명한 방법으로 (b)와 같이 로그 파일을 기록한다[1].

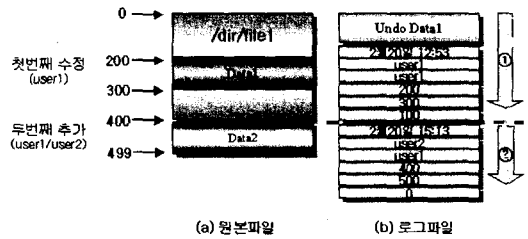


그림 2: 침입 복구를 위한 로그 저장 기법

그림 2에서 첫 번째 user1이 특정 파일을 수정을 하게 되면 수정되기 전 데이터는 로그파일에 기록되고 시간정보와 사용자 정보, 변경전 내용(Undo Data)에 대한 메타정보가 기록된다. 두 번째 user1에서 user2로 사용자 변경을 하고 내용을 추가하면 (b)의 ②와 같이 User Id와 Real User Id가별도로 기록된다.

* 본 연구는 대학 IT 연구센터 육성/지원사업의 연구결과로 수행되었음

2.3 로그 퍼지 기법

침입 복구 기법을 위한 로그파일의 크기는 계속적으로 증가한다. 로그파일 크기의 증가는 저장 공간의 문제를 일으킬 뿐만 아니라 복구시 많은 로그를 검색하여야 하므로 성능의 문제가 발생한다. 이러한 문제를 해결하기 위하여 신뢰된 구간에서는 로그 퍼지 기법을 사용하여 로그를 관리한다.

그림 3의 (a)와 같이 로그파일에 기록된 로그는 복구할 때만 활용되도록 생성되었기 때문에 특정 시점 이전의 로그를 삭제하는 것으로 간단하게 퍼지 기법을 구현할 수 있다.

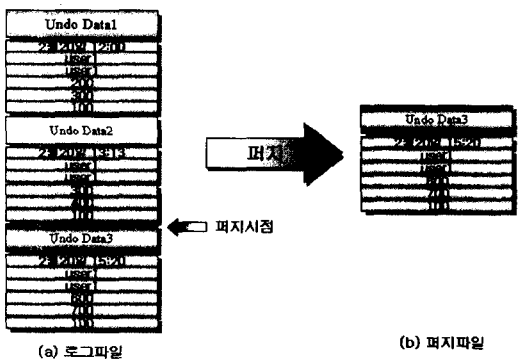


그림 3: 침입 복구를 위한 로그 퍼지 기법

2.4 침입 복구 기법

로그에 기록된 시간 정보를 활용하여 신뢰성 있는 부분의 로그인지를 판단하고, 비 신뢰성을 갖는 로그에 대하여서는 로그의 역할 정보를 활용하여 복구의 대상이 되는지 여부를 판단한다.

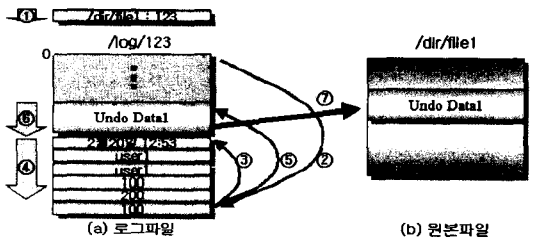


그림 4: 침입 복구 기법

그림 4는 침입 복구 기법을 나타낸다. 일련의 과정은 다음과 같다.

① 설정파일 리스트를 통하여 원본파일의 inode번호로 생성된 로그파일을 찾는다. 로그파일이 존재하면 로그파일을 연다.
② 옵셋 0의 위치에서 파일의 끝으로 옵셋을 이동한다.
③ 로그 헤더 정보를 획득하여
④ 침입복구 할 것인지를 시간 정보와 역할 정보를 이용하여 판단한다. 만약 복구가 필요하다면,
⑤ 로그파일의 끝으로 포인터가 이동하였으므로 변경전 내용을 획득하기 위하여 내용위치로 이동하고,
⑥ 변경전 내용인 Undo Data1를 읽어서
⑦ 원본파일에 적용한다.
⑧ ③~⑦의 과정을 복구원도우[1] 범위 내에서 반복하여 손상된 원본파일을 복구한다.

3. 침입 복구 모듈의 구현

그림 5는 침입 복구 모듈이 적재된 후의 시스템 구성도이다. 기존의 시스템 호출을 가로채어 일반적인 시스템 호출이 하는 동작 외에 침입 복구를 위해 특정파일 변경시 로그파일을 사용자에게 투명한 방법으로 기록하는 작업과 증가하는 로그를 관리하기 위해 로그 퍼지 기법을 수행하고, 침입이 발생하여 파일시스템이 훼손될 경우 침입 이전으로 복구하기 위한 동작을 수행한다. 이런 모듈 프로그램을 위해 로그 대상 파일에 대한 자료 구조와 원본파일기술자와 로그파일기술자의 사상을 위한 자료구조를 정의한다.

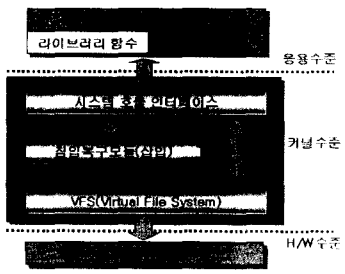
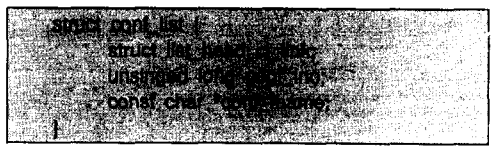


그림 5: 침입 복구 모듈 적재 시스템 구성도

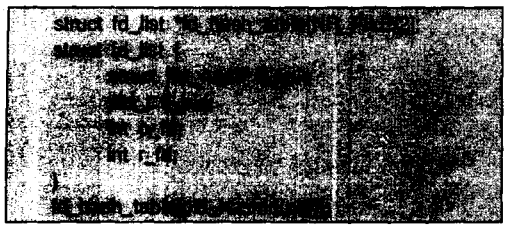
3.1 로그 대상 파일 리스트

침입 복구 모듈은 시스템 내의 모든 파일의 변경에 대하여 로그파일을 생성하는 것이 아니라 선택적으로 로그파일을 생성하여 복구에 이용한다. 특정파일은 다음과 같이 inode 번호(conf_ino)와 파일의 전체경로이름(conf_fname)을 갖는 이중 연결 리스트로 구현한다.



3.2 원본파일기술자와 로그파일기술자의 사상

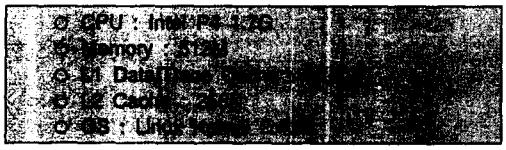
파일을 변경하기 위하여 모든 시스템은 sys_open() 시스템 호출을 하게 된다. sys_open() 시스템 호출은 파일기술자를 반환하게 되는데, 이때 사용자에게 투명한 방법으로 로그파일의 기술자도 반환해 주어야 sys_write()와 sys_close()에서 사용가능하다. 모듈은 원본파일기술자와 로그파일기술자를 사상하기 위하여 다음과 같은 리스트 구조를 갖는다.



또한 리눅스 시스템에서 파일기술자는 프로세스마다 256개씩 할당되며 시스템 내에는 1024개의 프로세스가 허용되기 때문에 해싱기법을 이용하여 사상을 구현하였다.

4. 오버헤드 분석

오버헤드 분석을 위해 시스템에 침입 복구 모듈을 적재하고, 로그를 관리하기 위하여 open(), write(), close() 시스템 호출시 발생하는 각각의 오버헤드를 분석한다. 오버헤드 분석을 위해 사용하는 시스템 사양은 다음과 같다.



4.1 open()/close() 오버헤드 분석

침입 복구 모듈을 적용할 경우 sys_open() 시스템 호출에서는 원본파일이 중복의 대상이 되는지 검색한다. 만약 중복 대상이 된다면 로그파일기술자를 생성하고, 생성된 파일기술자를 사상한다. sys_close() 시스템 호출의 경우에는 로그파일기술자를 원본파일기술자와 함께 반환한다. 그림 6은 open()과 close()에 수반되는 오버헤드를 나타낸다.

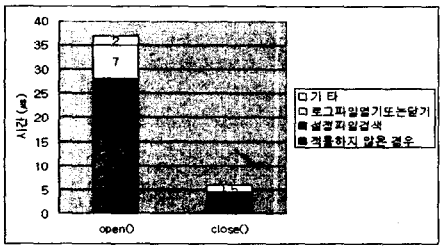


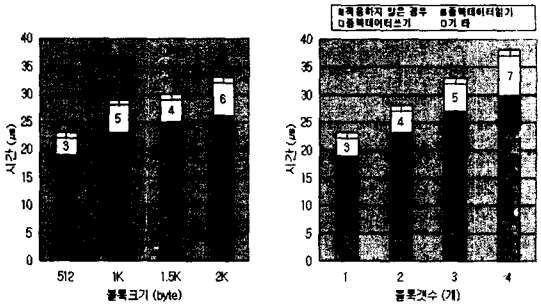
그림 6: open()/close()의 오버헤드 분석

설정 파일 리스트에서 중복 대상이 되는지 여부를 검색하는데 많은 오버헤드가 수반되는 것을 확인할 수 있다. 그림 6에서 설정 파일 리스트에서의 검색 시간은 평균 10번째로 중복 대상 파일이 발견될 때를 가정하고 시간을 측정한 결과이다.

4.2 write() 오버헤드 분석

그림 7의 (a)는 데이터의 크기(블록크기)를 다르게 하여 원본파일을 변경할 경우 write()에 수반되는 오버헤드를 분석한 결과이다. 원본파일을 write()하기 전에 원본파일의 변경이 일어나는 부분의 데이터를 읽어 로그파일에 변경전 데이터를 기록하고 로그에 대한 메타정보를 기록하는 시간이 추가적으로 발생한다. 일반적인 write()를 수행하는 시간의 53.3%~65%가 로그 파일의 기록을 위한 오버헤드이다.

그림 7의 (b)는 가변 크기의 블록 대신 512 byte의 고정 크기를 갖는 데이터 블록을 원본파일에서 1~4회 수정할 경우 write()에 수반되는 오버헤드를 분석한 결과이다. 이 경우에도 오버헤드의 크기는 블록의 크기가 가변적인 경우와 크게 다르지 않았다.



(a) 블록크기에 따른 오버헤드 (b) 블록갯수에 따른 오버헤드

그림 8: write()의 오버헤드 분석

5. 결론 및 향후 연구 방향

본 논문에서는 리눅스 파일시스템에서 로그 기반 침입 복구 모듈을 구현하였고, 모듈을 적재할 경우 발생하는 오버헤드를 측정하였다. 사용자에게 투명한 방법으로 원본파일의 변경에 로그파일을 생성하고 관리하기 위하여 발생하는 오버헤드는 적당한 수준으로 판단된다.

앞으로 로그 대상 파일을 검색하는 메커니즘을 개선하여 침입 복구 모듈의 오버헤드를 감소시키기 위한 연구와 생성된 로그파일에 대한 보안성을 보장하기 위한 연구를 진행할 계획이다.

참고문헌

[1] 이재국, 김형식, "리눅스 파일시스템에서의 로그 기반 침입 복구 기법", 한국정보과학회 2003년 봄 학술발표논문집 (A), pp. 413-415, 2003년 4월.

[2] Daniel P. Bovet and Marco Cesati, *Understanding the LINUX KERNEL*, pp. 303-420, O'REILLY, Dec. 2002.